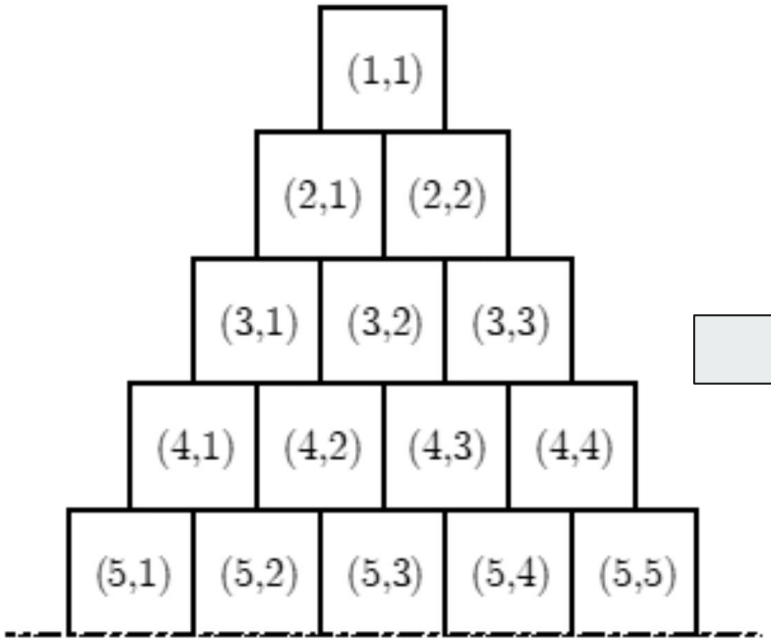
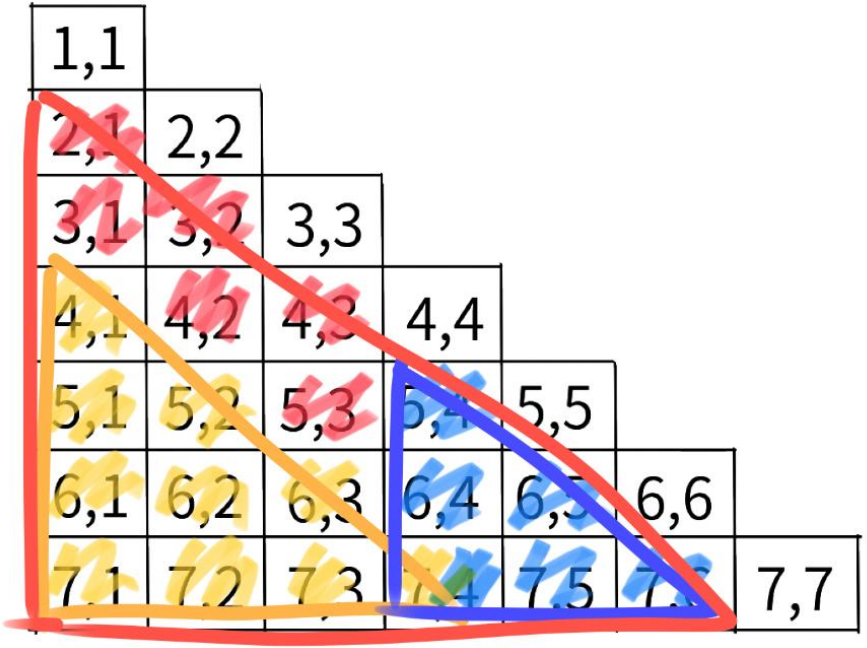


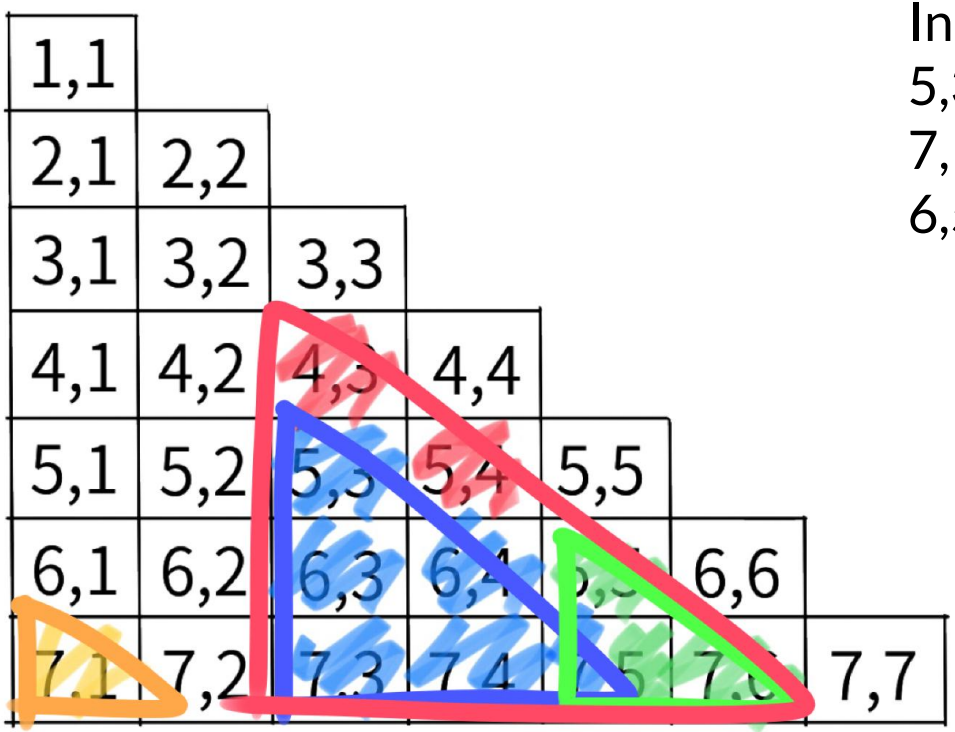
1,1				
2,1	2,2			
3,1	3,2	3,3		
4,1	4,2	4,3	4,4	
5,1	5,2	5,3	5,4	5,5

1,1						
2,1	2,2					
3,1	3,2	3,3				
4,1	4,2	4,3	4,4			
5,1	5,2	5,3	5,4	5,5		
6,1	6,2	6,3	6,4	6,5	6,6	
7,1	7,2	7,3	7,4	7,5	7,6	7,7

1st requirement



2nd requirement



Input:
5,3
7,1
6,5

```cpp
#include <iostream>
#include <algorithm>
#include <utility>
using namespace std;

pair<long long,long long> bricks[1001];

int main() {

  long long n,k;
  cin>>n>>k;
  for(long long i=0;i<k;i++){
    cin>>bricks[i].second>>bricks[i].first;
  }
  sort(bricks,bricks+k);

  long long ans=0;
  long long x=bricks[0].second,y=bricks[0].first;

```

```
19
20    for(int i=1;i<k;i++){
21
22      if(bricks[i].first-y-1<=n-x){   //checks if the 2 triangles touch each other
23
24        if(bricks[i].second>x && bricks[i].first>=y && bricks[i].first<=y+abs
            (bricks[i].second-x)){ //checks whether the block lies inside the triangle or not
25        }else{
26          x=bricks[i].second-(bricks[i].first-y); //turns the 2 triangles into 1 big one
27        }                                          //x=highest point of big triangle
28
29      }else{
30        ans+=(1+n-x+1)*(n-x+1)/2; //頭加尾乘項數除二
31        x=bricks[i].second;
32        y=bricks[i].first;
33      }
34    }
35
36    ans+=(1+n-x+1)*(n-x+1)/2;   //頭加尾乘項數除二
37    cout<<ans-k;  //minus the number of black blocks that aren't painted
38
39  }
40
```